Aalborg Universitet

AALBORG UNIVERSITY
DENMARK

Finding Similarity Functions for Classification with Genetic Programming

Preliminary Results

Guzman-Trampe, Juan; Cruz-Cortes, Nareli; Ortiz-Arroyo, Daniel

Published in:
Proceedings of EVOLVE 2012

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Guzman-Trampe, J., Cruz-Cortes, N., & Ortiz-Arroyo, D. (2012). Finding Similarity Functions for Classification with Genetic Programming: Preliminary Results. In C. Coello-Coello, & O. Schuetze (Eds.), *Proceedings of EVOLVE 2012: A Bridge between Probability, Set Oriented, Numerics, and Evolutionary Computation {II}*

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: May 04, 2023

EVOLVE 2012
Mexico City, Mexico, August 07-09 2012
ISBN 978-2-87971-112-6 ISSN 2222 - 9434

Finding Similarity Functions for Classification with Genetic Programming: Preliminary Results

Juan Evencio Guzman-Trampe[a], Nareli Cruz-Cortés[a] and Daniel Ortiz-Arroyo[b]

[a]Centro de Investigación en Computación
del Instituto Politécnico Nacional (CIC-IPN), Mexico [1]
gtrampea10@sagitario.cic.ipn.mx,nareli@cic.ipn.mx

[b]CISLab, Department of Electronic Systems
Aalborg University, Denmark
do@es.aau.dk

Abstract

In this paper we propose a Genetic Programming algorithm designed with a coevolutive scheme for classification problems. Our algorithm searches for similarity functions that are applied to compare pairs of objects from a supervised sample. The output of these functions can be used in similarity-based classifiers.

1 Introduction and Related Work

Classification is a machine learning problem that basically consists in determining the class of an object given its features. In supervised classification, the goal is to determine classes of objects from patterns learned from a training labeled set. A classifier trained using supervised learning, generalizes the learned patterns and applies them to unseen unlabeled objects. Contrarily, in unsupervised learning, classifiers do not require a training set; objects are placed together with other objects with whom they share more similarity. Many classification methods have been proposed in the literature using a wide variety of techniques such as Artificial Neural Networks, Bayesian Networks, decision trees, and K-Nearest Neighbor (KNN) or by using a combination of some of these techniques [1].
Genetic Programming (GP), like Genetic Algorithms is inspired by Darwin's natural evolution theory. GP codifies population individuals that represent a solution to a problem (or a part of it), using complex language representations.
In *coevolutive models* in GP, two or more populations evolve using cooperative or competitive schemes. In cooperative schemes each population represents a part of the problem to be solved. Therefore, together these populations solve problems using a divide and conquer approach. Contrarily, in competitive schemes, individuals compete against each other, with the goal of obtaining the best fitness value, at the expense of discarding other candidate solutions.
Distance functions are used in classification to determine how close two patterns lie within a predefined metric space. *Similarity Functions* are also used for the same purpose, but they have the advantage of being easier to generate since they have to satisfy only partially the properties of distance functions.
In general classification tasks can be broadly divided into three groups: preprocessing, model extraction, and post-processing. GP has been applied in all these tasks [2].
One of the most important *preprocessing* tasks in classification is feature selection. In feature selection, the vector containing the original features F of an object is transformed into another vector F' with a reduced number of features that either, maximizes or is at least as good as F with respect to some criterion. GP has been applied in feature selection to reduce the dimensionality of the problem by selecting the features that are most relevant to define classes [3].

[1]The authors acknowledge the support of SIP-IPN through project 20120002

In the *model extraction* task each individual solution generated by GP represents either a classifier, or a part of it. Using evolution the classifiers that show the highest quality are selected. GP has been used in this case to evolve for instance classification rules, discriminant functions, or decision trees [4]. For example, in [5] is proposed a GP to generate a classifier, where the individuals are mathematical expressions that combine features and coefficients. The coefficients are integer values, and the features are characters indicating their position in the pattern's description. A similar work can be found in [6] where the aim is to predict whether a company increases its stock value compared with the previous year, the individuals are logical rules, instead of the mathematical expressions.
Post-processing tasks comprise several techniques, including *ensembles of classifiers*. In this approach, a group of classifiers is combined with the goal of producing together better results. GP has been utilized in this case to either optimize the base classifiers used in the ensemble or to combining different classifiers in the most optimal way [7].
Classifiers can be constructed using searching techniques to find functions that could be used to calculate the degree of similarity among objects' features (a technique called *partial similarity*) or by searching for similarity functions that could be used to compare whole objects (known as *complete similarity*).
This work uses GP together with a cooperative coevolutive scheme to search for partial similarity and complete similarity functions. The functions found by our approach can be used as inputs to similarity-based classification algorithms such as K-NN, or others.
This document is organized as follows, in Section 2 we provide some definitions and describe the problem statement. Section 3 presents our approach with some detail. Section 4 describes the experiments and results obtained. Finally, in Section 5 we state our conclusions.

2 Definitions and Problem Statement

A supervised classification problem SCP can be defined as $SCP = \{O, R, C\}$ where O is a set of training patterns, and $(|O| = n)$ is the number of elements in the traning set, R is the set of features that define the objects, each with a domain $d_k = \{r_k^1, \cdots, r_k^n\}$, and C is the set of tags representing the classes.
Let O_i and O_j be two objects in O. We are interested in finding the *partial similarity function* for each k-th feature

$$f_k(O_i, O_j),\ \forall k \in |R|,$$

that best describes the similarity between the two objects i, j regarding feature r:

$$f_k : d_k \times d_k \rightarrow \Re$$

Additionally, the *complete similarity function*:

$$F(O_i, O_j) = (f_1(O_i, O_j), \cdots, f_k(O_i, O_j), \cdots, f_r(O_i, O_j))$$

denotes the similarity degree between two objects O_i and O_j. The similarity functions must fulfill the following characteristic:

$$F : \Re^n \times \Re^n \rightarrow \{F_{min}, F_{max}\} \text{ where } F_{min}, F_{max} \in \Re$$

$$F(O_i, O_j) = \begin{cases} F_{max} & \text{If } C(O_i) = C(O_j) \\ F_{min} & \text{Otherwise} \end{cases}$$

Where F_{min} and F_{max} are the lowest and the highest possible values of F respectively, and $C(O)$ is the class of the object. F_{max} is obtained by comparing objects of the same class, and F_{min} is obtained when the objects belong to different classes.

3 Proposed Approach

We use GP to find the similarity functions that best describe the relationships between *features* and *objects* in a supervised sample. These functions can then be used as input to similarity-based classifiers, such as K-Nearest Neighbors (KNN).
To find the similarity functions we use a coevolution scheme splitting the problem into two parts. Firstly, the partial similarity functions between pairs of features are searched for. Secondly, the complete similarity functions are evolved and those individuals having the best fitness value are selected. This last step includes also a way in which the partial functions must be combined.
To illustrate the procedure, consider two populations **A** and **B** that will coevolve. The population **A** will search for the relationships between complete objects given the population **B** i.e. the *complete similarity functions* will be generated. The second population **B** searches for the relationships between single features i.e. the *partial similarity functions* will be generated.
In both populations the parent and survivor selection and variation operators (crossover and mutation) are independently applied but they both are used to compute fitness values.
The overall procedure can be described as follows:

1. Execute the GP algorithm to learn the partial and complete similarity functions based on the training sample.

2. For an unknown pattern, compute the similarity values using the similarity functions.

3. Apply a similarity-based classifier algorithm.

3.1 Individuals Representation

The individuals in GP are represented with tree structures. Our scheme uses two populations, **A** and **B**, where **A** will evolve trees representing the relationships between complete objects. The individuals in population **A** are composed by operators (+, -, *, ÷) and terminals (the partial functions f_1, ... , f_r, where r is the number of features defining the objects) additionally to the integer numbers into the range [0,100]. The reason to select integer numbers is to limit the possibilities, and the range was obtained experimentally.
Each individual in **B** is composed by r trees representing the features. These individuals are conformed by operators (+, -, *, ÷) and terminals that correspond to the k-th feature values of one pair of objects (O_i, O_j). This is denoted as X_k^m and X_k^M where the superscript refers to the minimum (m), or to the maximum (M) of both values. Additionally, the integer numbers in range [0,100] are also terminals. These representations are illustrated in Figures 1 and 2.

3.2 Fitness Function

In a coevolutive scheme, the fitness value of individuals in population **A** depends on the individuals in population **B**, and vice-versa. Using the individual representations of populations **A** and **B**, it is possible to obtain similarity functions completely defined, if one individual from population **A** is combined with one individual from population **B**. Thus, by combining an individual a_i with every other individual from **B**, we will obtain $|B|$ completely defined similarity functions. These $|B|$ similarity functions, are used to calculate the fitness of the individual a_i by counting the objects that are correctly classified when applying

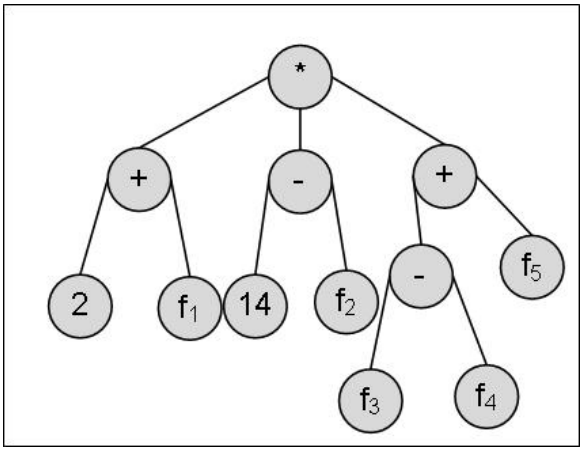

Figure 1: Example of an individual of the population **A**

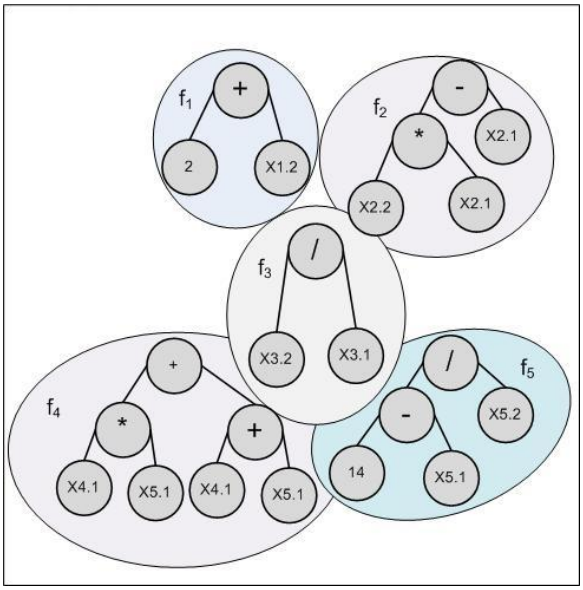

Figure 2: Example of an individual of the population **B**

a similarity-based classifier to the training set O. The classification algorithm takes as input all the $|B|$ similarity values.
Assuming that we have the following training set of objects O_i:

$$O_1 = \{1, 1, 1, 3 : 1\}$$
$$O_2 = \{1, 1, 2, 2 : 1\}$$
$$\vdots$$
$$O_{24} = \{2, 2, 2, 3 : 3\}$$

Then, an individual a_i in population **A** (a complete function) could be formed with the operators and the terminals $\{f_1, f_2, f_3, f_4\}$ additionally to integers in the range [0,100]. For instance, two possible individuals a_1 and a_2 could be generated as:

$$a_1 = \frac{f_1 * (f_2 - f_3)}{f_4 * (f_1 + 5)} \tag{1}$$

and

$$a_2 = f_2 * f_4 + (88 - f_3) + f_1 \tag{2}$$

On the other hand, an individual b_j in the population **B** is formed by four trees f_1, f_2, f_3, and f_4 where each f_k refers to the k-th feature. For instance, the tree f_1 may contain the values that some object's feature (like *age* for example) could take. An individual b_1 is represented by:

$$b_1 \begin{cases} f_1 = & X_1^M - X_1^m \\ f_2 = & (X_2^M)^2 - X_2^m + (X_2^M / X_2^m) \\ f_3 = & X_3^m + X_3^M \\ f_4 = & X_4^M * X_4^m \end{cases}$$

The fitness value of an individual a_1, is calculated by combining it with each other individual in population **B** to obtain $|B|$ complete similarity functions. These similarity functions are then used to classify the training set O. The fitness value is the number of correctly classified objects.
An example of how one completely defined similarity function is obtained by combining a_1 with the individual b_1 of **B** is shown next,

$$a_1 = \frac{(X_1^M - X_1^m) * (((X_2^M)^2 - X_2^m + (X_2^M / X_2^m)) - (X_3^m + X_3^M))}{(X_4^M * X_4^m) * ((X_1^M - X_1^m) + 5)} \tag{3}$$

4 Experiments

To validate our proposal, we performed a series of experiments. The test datasets were taken from the UCI Machine learning Repository[2]. Table 1 contains a description of the *Pima Indian Diabetes, Haberman's Survivors, Monks Problem, Breast Cancer Wisconsin (Diagnostic), and Blood Transfusion Service Center* data sets.
In our experiments 80% of the data set was randomly selected as the training set, and the remaining 20% utilized in the validation phase. The parameters values applied to the experiments are shown in Table 2:
Our experiments consisted in applying the proposed GP to find the similarity functions to the five test datasets. The K-NN algorithm was utilized to perform the classification to obtain the fitness values. The obtained similarity functions were applied to the validation dataset. The percentage of correctly classified objects from 20 independent runs is presented in Table 3.
In addition to that, we applied an Euclidean distance to the K-NN algorithm to the same test datasets. The results obtained are shown in Table 4. Thus, from these we can observe that our results are similar to the Euclidean distance ones.

[2]http://archive.ics.uci.edu/ml/datasets.html

Dataset	Objects	Features	Classes	Attribute types
Pima Indian Diabetes	768	8	2	Integer and real
Haberman's Survivors	306	3	4	Integer numbers
Monks Problems	432	7	2	Categorical
Breast Cancer Wisconsin	569	30	2	Real numbers
Blood Transfusion Service Center	748	5	2	Real numbers

Table 1: Test data sets

Parent Selection	Roulette Wheel
Crossover Rate	0.9
Mutation Rate	1/(Number of leafs in the tree)
Size of Population A	20
Size of Population B	30
Number of Generations	400

Table 2: Parameter values used in the experiments

Dataset	Worst (%)	Best (%)	Average	Std. Dev.
Pima Indian Diabetes	67.97	74.15	70.66	2.69
Haberman's Survivors	80.85	91.24	87.36	3.25
Monks Problem	70.75	74.54	72.59	1.8
Breast Cancer	72.21	75.8	75.10	0.48
Blood Transfusion Service Center	70.2	72.21	71.00	0.6

Table 3: Statistical results from 20 independent runs of the proposed method.

Dataset	Worst (%)	Best (%)	Average	Std. Dev.
Pima Indian Diabetes	67	67	67	1.29
Haberman's Survivors	73	75	74	1.08
Monks Problem	73	80	78	1.8
Breast Cancer	93	93	93	0
Blood Transfusion Service Center	72	72	72	0

Table 4: Results of the 20 runs of the euclidean classifier.

5 Conclusions and Future Work

We proposed a GP algorithm that uses a coevolutive scheme to find partial and complete similarity functions. The partial similarity functions find similarity values between pairs of object's features. The complete similarity functions find similarity values between objects by combining the partial functions to calculate the similarities of complete objects. Our results show that our approach is able to find these complete similarity functions. These functions are used to compare objects' similarities for classification. Our experiments show moderately competitive results when compared to other similar approaches. However, these results are preliminary. It is necessary to conduct more experiments so that the parameters of our model could be tuned optimally and find other fitness functions that we could use, together with testing the similarity functions found by our approach with other similarity-based classifiers.

References

[1] Miqueles Teresa, Bengoetxea, and Larrañaga Pedro. Evolutionary computation based on bayesian classifiers. *International Journal On Applied Mathematics and Computer Science*, 14(3):335–349, 2004.

[2] P.G. Espejo, S. Ventura, and F. Herrera. A survey on the application of genetic programming to classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(2):121 –144, March 2010.

[3] Guo H., Jack LB, and Nandi AK. Feature generation using genetic programming with application to fault classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35:89–99, 2005.

[4] Mitsuyoshi Shirasaka, Qiangfu Zhao, Omar Hammami, Kenichi Kuroda, and Kazuyuki Saito. Automatic design of binary decision trees based on genetic programming. In *Proc. The Second Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'98*, 1998.

[5] Kishore J.K., Patnaik L. M., Mani V., and Agrawal V. K. Application of genetic programming for multicategory pattern classification. *IEEE Transactions On Evolutionary Computation*, 4(3):242–258, 2000.

[6] Doherty Gregory. Fundamental analysis using genetic programming for classification rule induction. In John Koza, editor, *Genetic Algorithms and Genetic Programming*, pages 45–51. Stanford Bookstore, 2003.

[7] William B. Langdon, S. J. Barrett, and B. F. Buxton. Combining decision trees and neural networks for drug discovery. In *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002, volume 2278 of LNCS*, pages 60–70. Springer-Verlag, 2002.